GamingFreedom client/server documentation

1 . Preface

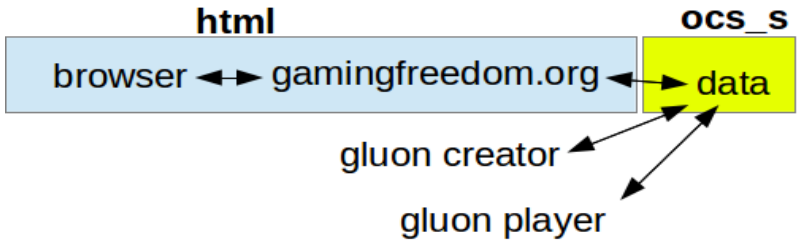
Gluon is a project that aims to deliver a complete framework for creating and playing videogames.
In order to give the player a great gaming experience we need to give him also an awesome distribution and feedback service. What's really missing is a cutting-edge website that satisfies players' desires like sharing statistics and games, discuss games with friends and much more. This can also help development allowing developers to receive feedback quickly and respond to it.

2 . Requisites

We will need mainly two parts for this project to work. First, we start defining two different parts of this project:

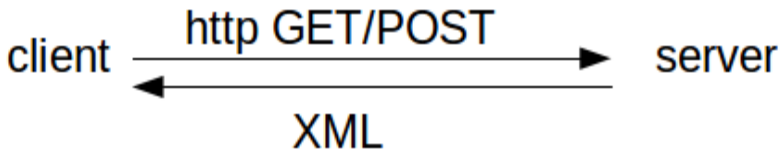
1. **html** → this part could be identified as the websites that resides on the address gamingfreedom.org and will be browsable with a standard web browser. This is what the final user really can see.
2. **data** → this constitutes in an *OCS server* that will really handle and modify data following OCS protocol definitions. This is the only part that could have direct access to its database (ocs data related).

Here is a simple graph that explains how those 2 big parts of this project are relationed.



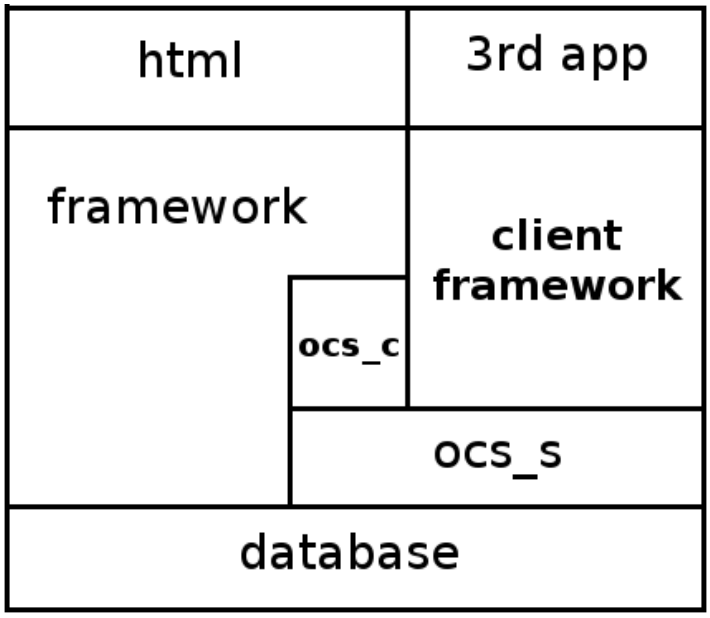
Note that the **html** only serve eventual web browser clients and that won't have direct access to the database (at least for data regarding this project). Any other kind of data that is just **html** dependant is allowed to be stored in the database in different tables or in a completely different database.
Also, **data** will serve any other third-part application implementing the OCS API and serving clients that are compatible with OCS protocol.

To go a little more in depth in how the structure of the entire project will be we have to consider how OCS works, as a RESTful protocol.



As specified in the OCS specification (as for now 1.6) **client** send an http request passing data using GET and POST methods to the server, which, once evaluated, respond with an http content, containing standard XML. Note that the XML has to be processed by the client directly.

In our case the **client** is **html** which send the needed requests to **data** (identifiable here as **server**). Given those requisites and basic structure, here is the structure a little more in depth that consider the OCS side of every element.



As mentioned, ocs_s (OCS server) will still have exclusive access to the database tables related to OCS data in order to maintain data integrity and stability. Every other 3rd app will talk through client framework at ocs_s. The client website will access OCS data directly making http get/post requests to itself.

> **Note:**
> Even if this is not optimal and efficient like making requests using the ocs_s client API, this will avoid breaking compatibility between http and ocs_s hosted *on the same machine* and *hosted on different machines*.

3 . Licensing

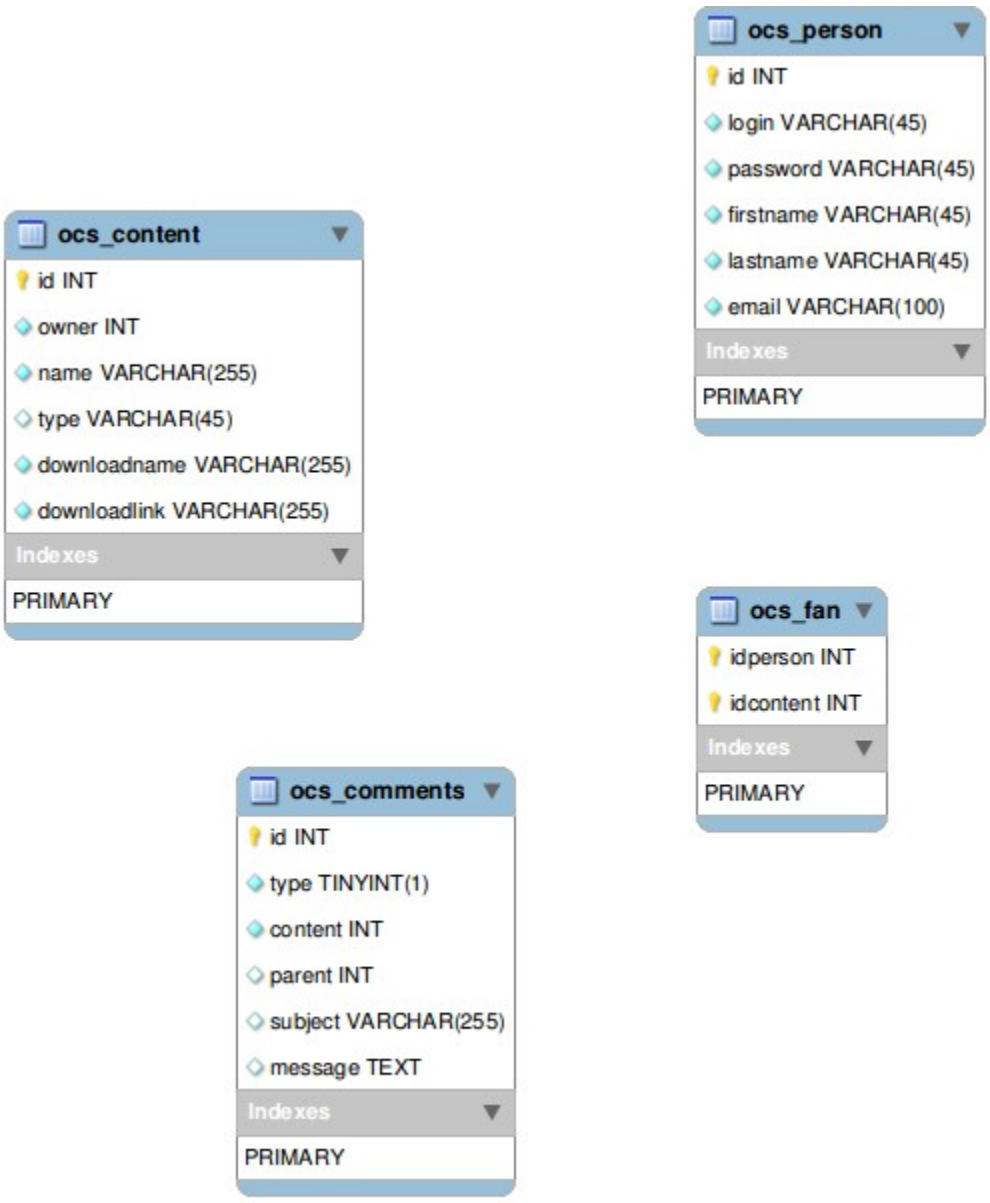
The resultant product must be opensource software, which means that has to be released under a compatible opensource license. Simple OCS Server will be released under the opensource license GPLv3.

4 . Language

Observing the type of project, with a strong relations to a REST interface for nearly every data exchange between server and clients we will give the assumption, for now, that the preferred language used will be **PHP** (www.php.net), for both ocs_s and http (gamingfreedom.org).

5 . Database

I decided to adopt MySQL for this project. This decision comes with the fact that the preferred php framework for this project is GFX3, and this late one mainly support MySQL. However, a multiple DB driver is in plan in order to increase compatibility with other database engines.
This is how the structure should look like, following the OCS specification:



[insert some other specs here]

6 . OCS Server Engine

Given those assumptions, we'll start building a working OCS server implementation. Not all the OCS protocol will be implemented, but we'll give support for the following:

- CONFIG
  - config
- PERSON
  - -all (every request callable)
- FAN
  - -all (every request callable)
- CONTENT
  - list
  - get
  - download

    - vote
    - add
    - edit
    - delete
    - upload download file
    - delete download file
    - upload preview image
    - delete preview image
- COMMENTS
    - -all (every request callable)

Since there is not a working framework capable of handling an ocs requests, we'll need to write one, possibly reusing already existant code from the example php implementation http://socialdesktop.org/library/lib_ocs.txt which is AGPL licensed. To make things easier we'll call this framework **SOCS** which stands for **Simple OCS Server**. The framework backend on which SOCS will be based is GFX3.

6 . OCS Client Engine

In order for **html** (gamingfreedom.org) to work we will use the GFX3 framework, which can be found at this link: www.gfx3.org. Effects 3 is convenient against other frameworks because it's easily moddable, it's flexible and an OCS client module is really easy to integrate. For the writing of this module, the reference OCS client will be reused.

GFX3 features of our interest:

- template engine
- modules
- user management
- low / high level api driver for mySQL
- cache engine

Our choice to choose GFX3 is mainly influenced by the fact the person working on this project knows well this engine and is the actual main contributor which means that code can be integrated and modded in an extremely easy way. This could be considered as a codebase from which we start working on those 2 projects.